

Issues Discovering Compromised Machines

Anton Chuvakin 2004-10-25

One of the latest security books I read had a fascinating example in the preface. The authors, well-known and trustworthy experts in the field http://wwwdev.securityfocus.com/cgi-bin/preview/infocus_preview.pl?id=1809 of security, made an outrageous claim that most of the Fortune 2000 companies have *already* been penetrated by hackers (and have been in that state for years!). Hackers move in and out at will through the backdoors and other covert channels without the security personnel knowing or even suspecting it. Without being able to verify the validity of this, I decided to look at the problem of reliably discovering the compromised machines on corporate networks. Reliability is of key importance here as there are lots of ways to obtain a *suspicion* that the machine is "owned" or infected, but sadly there are few truly reliable ways to discover that short of full forensic analysis, likely requiring physical access to a machine as well as shutting it down for a potentially long time.

In addition, as advanced blackhat community moves beyond buffer overflows into new exploit type areas and zero-days attacks (with non-public exploits against non-public vulnerabilities) have a chance of becoming more common, traditional intrusion detection rates might decrease even further, giving defenders no chance to detect, let alone stop the attack. Obviously, in case of a zero-day attack, detecting a second order (i.e. after-exploit) trace becomes the only option, as the attack itself will most likely fly through most network security monitoring gear by the very nature of being a "zero-day". Some of the existing attacks can also be mutated and optimized to pass through some of the detection filters.

First, let us outline what we mean by a "compromised" machine. A machine is considered "compromised" if:

- It was successfully exploited and used by attackers
- It has a backdoor or other malicious software running without the knowledge of the system owner
- It was configured to allow access to unauthorized parties without the owner knowing it

Here are some examples of the above that most people running real-world networks can identify with:

Example 1 A system is infected by a worm and then starts sending out humongous amounts of

malicious traffic, trying to spread the infection in the organization's network and beyond

Example 2 System access privileges on the exposed FTP server are abused and it is then taken over by amateur attackers to store and share their stash of porn and pirated software

Example 3 A server is exploited by attackers and a rootkit is deployed. Several system binaries are trojaned for easier and more covert attacker access. Also installed is an IRC bot that "calls home" to a specific channel for commands.

The above example might give the reader some hints on how to detect a compromise in those specific cases, but we will treat the problem in a more general fashion.

Let us first look at the current technologies used for detecting the activities of the attackers. There are attack detection systems (often mislabeled as "intrusion detection") that look for various attack *attempts* and probes. Then, there are intrusion detection systems that look for known intrusion methods utilized by hackers. Here, however, we are talking about reliable compromise detection: detecting the machines that were attacked, intruded and are now used by either human attackers or their automated aides, virus-like code known as malware.

As noted above, reliability of detection is critical since the stakes here are much higher than in attack/intrusion detection; the alerts we will raise will be immediately actionable (unlike what is coming from most deployed NIDS today). It will be, "your machine was hacked and is now used by attackers, run there and do **something**," instead of, "it seems like somebody might be attacking you or something of that sort, you might want to pay attention."

How can we detect that the machine is indeed compromised? The answer heavily depends upon what kind of information and tools are available, as well as the position of the user. It is one thing to only go by the firewall data relevant to the potentially compromised machine and it is completely different if we have full access to the hardware, OS and application for such system in the forensics environment. Realizing the benefits from the latter case obviously requires access to a skilled computer forensics professional as well as significant time investment.

To get some background let's briefly review some common activities occurring on the systems compromised by attackers. These signs were discovered by the author over the years of the honeypot research with the HoneyNet Project as well as by other related efforts. Obviously, they refer to activities by the outside attackers and not by malicious insiders.

1. A backdoor daemon can be launched by an attacker on a high-numbered port or an existing daemon is trojaned. This is observed in most compromise cases.
2. Installing an IRC bot or bouncer is a popular choice. Several IRC channels are dedicated entirely for communication with the servers compromised by a particular group, and have been observed on numerous occasions
3. In this age of DDoS and reflexive DoS, the classic point-to-point DoS tools are not laid to rest. They are commonly used on some compromised machines to deal quick damage against the opponents on slower connections
4. Many attackers will use the compromised system for vulnerability scanning and widespread exploitation. Used scanners and auto-rooters are capable of discovering and exploiting massive (thousands and more) numbers of systems within a short time period. Such large networks can be, and are being used for devastating denial of service attacks
5. Using a compromised machine, finding pirated software or media is common as well. On high bandwidth servers, people are known to store gigabytes of "stuff" with active download and uploads occurring all the time. It is worthwhile to note that one is often not required to exploit the server to use it for storage, since enough servers are misconfigured with extra privileges given to all connecting users.
6. "Earning" some cash by getting involved in the stolen credit card trade (see <http://www.honeynet.org/papers/profiles/cc-fraud.pdf> for reference) is a new choice among nefarious hackers, rapidly growing in popularity. Such activity utilizes IRC with specially modified "trading" bots
7. Another money-making "venture" is sending spam or "loaning" a compromised box to spammers for a reasonable fee.
8. Another ominous activity is using the compromised machine for a "phishing" site. Attackers were observed deploying a fake bank site and then launching waves of spam emails attracting visitors who are quickly separated from their cash.

In light of the above, we can summarize some of the compromise detection methods, given different available resources (both technical and human). The last column in the tables below indicates detection reliability in the *absence* of any other information, ranging from highly uncertain ('low') to immediately actionable ('high'). This subjective rating, originating in author's experience with compromised systems, still provides a useful metric of how much trust should be put into each event.

We will start from simple compromise-related events and indicators.

Security technology/ resource	Method	Example	Reliability
NIDS	Compromise signature	Shell commands on SSL port TCP 443	Medium
NIDS	Post exploit activity	'whoami' in command flow	Medium
NIDS	Volume of outbound exploits (same or different)	Lots of SSL hits out	Medium
NIDS	Volume of outbound exploits after a similar inbound exploit	Lots of SSL hits out after the system is hit by SSL exploit	High
NIDS, firewall	Outbound massive port scanning, DoS, etc	Many connections to port 1434 UDP from a single system	Medium
HIDS	Abuse-related system log records	New account created	Medium
HIPS	Application behaving significantly different from known good	Connections, registry access, file replacements	Medium

The above provides a surprising conclusion: even if your system spews forth malicious exploits, everything might be okay and no risk may actually exist. The reason is simple: 'false alarms' (and, specifically, their most famous variety, 'false positives'). For example, it is rather common to see a corporate HTTP proxy generate an inordinate number of web exploits (such as long URLs, suspicious characters and strings in URLs, etc) just because a large number of web users try to access various web sites.

Now we will look at other, more circumstantial and complex patterns:

Security technology/ resource	Method	Example	Reliability
--	---------------	----------------	--------------------

Firewall, router, switch	Connections to new ports on the system	Multiple systems connecting to port TCP 12345 on the same box	Medium
Firewall, router, switch	New connections or attempts from machine to outside	IRC (TCP 6667) connections from system	High
Vulnerability scanner	New open ports (known backdoors)	12345 TCP open	Medium
Network monitoring system	Unusual connectivity from an internal system to outside	Lots of connections to port 6667 TCP to a set of servers from a single internal system	High
Network monitoring system	An unusually high volume of traffic from an internal system	A SYN flood from an internal system	Medium
Anti-virus or anti-spyware software	A warning message about a Trojan that cannot be cleaned or stopped	Backdoor.IRC.Bot running	High

Note that even anti-virus solution can give you false positives. Thus, even the supposedly unambiguous Trojan or backdoor alert from a AV software might well be a false alarm.

It is worthwhile to note that some of the above are facilitated by correlating NIDS data with other kinds of data (such as firewall or host data, or using a different NIDS) and thus arrive with the conclusion that the target is compromised via automated rule-based correlation. Also, correlation technology can automate the investigative process by looking at other related activities occurring at the time of the attack. As a result, we arrive at higher compromise detection quality than using the output of a single NIDS sensor. However, if one is lacking a good correlation engine, the analyst can still perform the same steps manually, even if not in real-time. A recent security book by Richard Bejtlich "Tao of Network Security Monitoring" covers some of the techniques.

We will now look at the most reliable indicators, with larger resource constraints:

Security technology/ resource	Method	Example	Reliability
System admin	New files, accounts, processes (see "SANS Intrusion Discovery Cheatsheet")	New 'rewt' account	High

More of such techniques are summarized in the SANS Intrusion Discovery sheets (available at www.sans.org for Linux and Windows systems). The techniques do require console access to a system, however, and thus are hard to scale to a large environment.

Now we will discuss how one can use the above methodology in production environments. The ideal way to apply most of the above guidance in an automated fashion is to set up a some kind of multi-vendor database with events, alerts, log records from various installed security devices and then to apply rules (also known as "correlation rules") across those events, either historically using stored events or in real-time across the incoming event feed. For example, one can easily look for events from a network IDS indicating exploitation attempts that are then shortly followed by multiple outbound connections (such as IRC, see above examples) as was as other compromise or worm-like events.

Commercial SIM (Security Information Management) products have that capability in the form of real-time correlation rules can also be useful. A budding open source product of the same kind, OSSIM (www.ossim.net) has started to develop this as well. OSSIM currently allows one to pair the attack event reports by the NIDS (such as Snort and others) with various attack responses, such as outbound communication in line with our methodology.

If one is lacking the budget for a commercial solution and the healthy dose of resolve that is sometimes needed for deploying an open source one, you can try building it yourself. The approach will be to combine the relevant events (such as Snort NIDS and a Linux IPTables firewall rules, for example, using the ACID schema as described in http://www.giac.org/practical/GSEC/Anthony_Shearer_GSEC.pdf). This should be done in the same database and then you apply the logic via a script or a query. The techniques can also be applied manually by looking through the logs, however this takes a significant amount of time.

To conclude, various techniques to discover compromised machines are a very handy tool to deal with attackers of all skill levels. Contrary to popular wisdom, it doesn't not always require expensive forensics services, but can be performed using collected network and system logs and other sources of information.

About the Author

Anton Chuvakin, Ph.D., GCIA, GCIH (<http://www.chuvakin.org>) is a Security Strategist with a major security company, where he is involved with designing the product, researching potential new security features and advancing the security roadmap. His areas of infosec expertise include intrusion detection, UNIX security, forensics, honeypots, etc. He is the author of a book "Security Warrior" (O'Reilly, January 2004) and a contributor to "Know Your Enemy II" by the Honeynet Project (AWL, June 2004) and "Information Security Management Handbook" (CRC, April 2004). In his spare time he maintains his security portal <http://www.info-secure.org>

View [more articles](#) by Anton Chuvakin on SecurityFocus.

[Privacy Statement](#)

Copyright 2006, SecurityFocus